UDACITY

‹ Back to Data Analyst Nanodegree

# Explore US Bikeshare Data

审阅

代码审阅 9

HISTORY

▼ bikeshare.py 9

```
1  time
2  pandas as pd
3  numpy as np
4
5 TA = { 'chicago': 'chicago.csv',
6         'new york': 'new_york_city.csv',
7         'washington': 'washington.csv' }
8
9 _filters():
10
11 s user to specify a city, month, and day to analyze.
12
13 urns:
14   (str) city - name of the city to analyze
15   (str) month - name of the month to filter by, or "all" to apply no month filte
16   (str) day - name of the day of week to filter by, or "all" to apply no day fil
17
18 nt('Hello! Let\'s explore some US bikeshare data!')
19 et user input for city (chicago, new york city, washington).
20 le True:
21   city = input('Would you like to see data for Chicago, New York, or Washington?
22   if city in ['chicago', 'new york', 'washington']:
```

建议

建议这里可以直接使用 `CITY_DATA.keys()` 来获取各个地区的名称，可以简化你的代码并减少不必要的冗余。

```
23       break
24   else:
25       print('Your answer is not correct, please try again.\n')
```

```
26 nt('Looks like you want to hear about %s! If this is not true, restart the prog
27
28 et user input for preferences of time filter
29 le True:
30  user_filter = input('Would you like to filter the data by month, day, both, or
31  if user_filter in ['month', 'day', 'both', 'none']:
```

棒极了

很好的使用了数组简化了判断流程，做的很好。

```
32      break
33  else:
34      print('Your answer is not correct, please try again.\n')
35 nt('We will make sure to filter by %s!\n\n' % user_filter)
36
37 onth can be assigned
38 th = 'all'
39 user_filter == 'month' or user_filter == 'both':
40  # get user input for month (all, january, february, ... , june)
41  months = {'january': 1, 'february': 2, 'march': 3, 'april': 4, 'may': 5, 'june
42  while True:
43      month_key = input('Which month? January, February, March, April, May, or J
44      if month_key in months:
45          month = months[month_key]
46          break
47      else:
48          print('Your answer is not correct, please try again.\n')
49  print('You will make sure to choose %s!\n\n' % month_key)
50 ay can be assigned
51 ' = 'all'
52 user_filter == 'day' or user_filter == 'both':
```

棒极了

很好的对逻辑判断的语句进行了简化，做的很棒。

```
53  # get user input for day of week (all, monday, tuesday, ... sunday)
54  days = {'sunday': 0, 'monday': 1, 'tuesday': 2, 'wednesday': 3, 'thursday': 4,
55  while True:
56      day_key = input('Which day? Sunday, Monday, Tuesday, Wednesday, Thursday,
57      if day_key in days:
58          day = days[day_key]
59          break
60      else:
61          print('Your answer is not correct, please try again.\n')
62  print('You will make sure to choose %s!\n\n' % day_key)
```

棒极了

很好的对输入进行了错误检测和处理以及容错处理，做的很棒。

```
63 nt('-'*40)
64 urn city, month, day
```

需要修改

是否发现了你的上方几处代码重复了很多，说明了你的代码可以进行重构，将不同的地方抽取出来作为函数的参数
现。这样将程序的逻辑封装起来，可以提高程序的可读性和降低程序的冗余以及减少程序的耦合。
例如你的代码可以参考下面的代码进行代码的重构。

建立一个用于获取输入的函数：

```
def get_item(input_print,error_print,enterable_list,get_value):
    while True:
        ret = input(input_print)
        ret = get_value(ret)#这里执行作为参数的函数
        if ret in enterable_list:
            return ret
        else:
            print(error_print)
```

你可以在get_filters函数中这样使用。

```
    city = get_item('Please input the city name:',
                    'Error!Please input the correct city name.',
                    ['chicago', 'new york city', 'washington'],
                    lambda x: str.lower(x))
    #如果是想使用title函数可以最后的函数传入 lambda x:str.title(x)
```

```
65
66
67 d_data(city, month, day):
68
69 ds data for the specified city and filters by month and day if applicable.
70
71 s:
72  (str) city - name of the city to analyze
73  (str) month - name of the month to filter by, or "all" to apply no month filte
74  (str) day - name of the day of week to filter by, or "all" to apply no day fil
75 urns:
76  df - Pandas DataFrame containing city data filtered by month and day
77
78 oads data for the specified city
79 :
80  df = pd.read_csv(CITY_DATA[city])
81 ept Exception as e:
82  print('The program encountered an error: ', e)
83  exit()
84
85 onvert the Start Time column to datetime
86 'Start Time'] = pd.to_datetime(df['Start Time'])
87 xtract month from the Start Time column to create an month column
88 'month'] = df['Start Time'].dt.month
89 xtract day from the Start Time column to create an day column
90 'day'] = df['Start Time'].dt.weekday
91 xtract hour from the Start Time column to create an hour column
92 'hour'] = df['Start Time'].dt.hour
93 xtract duration from the Start Time column and the End Time column to create an
94 'End Time'] = pd.to_datetime(df['End Time'])
95 'duration'] = df['End Time'].dt.date - df['Start Time'].dt.date
```

建议

很好的对时间进行了计算，做的好。不过时间的列在源数据中是存在的，为`Trip Duration`。而且其数据还是较低。

```
96
97 ilter some rows based on user's filter
98 month == 'all' and day == 'all':
99  return df
100 f month == 'all' and day != 'all':
101  return df[df.day == day]
102 f month != 'all' and day == 'all':
103  return df[df.month == month]
104 e:
105  return df[(df.month == month) & (df.day == day)]
106
107 e_stats(df, month_is_all, day_is_all):
108 Displays statistics on the most frequent times of travel."""
109
110 nt('\nCalculating The Most Frequent Times of Travel...\n')
111 rt_time = time.time()
112
113 month_is_all:
114  # display the most common month
115  popular_month = df['month'].mode()[0] # find the most popular month
116  month = ['', 'January', 'February', 'March', 'April', 'May', 'June']
117  print('Most popular month: %s\n' % month[popular_month])
118
119 day_is_all:
120  # display the most common day of week
121  popular_day = df['day'].mode()[0] # find the most popular week
122  day = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Satu
123  print('Most popular day of week: %s\n' % day[popular_day])
124
125 lisplay the most common start hour
126 ular_hour = df['hour'].mode()[0] # find the most popular hour
127 nt('\nMost popular hour: %d\n' % popular_hour)
128
129 nt("\nThis took %s seconds." % (time.time() - start_time))
130 nt('-'*40)
131
132
133 tion_stats(df):
134 Displays statistics on the most popular stations and trip."""
135
136 nt('\nCalculating The Most Popular Stations and Trip...\n')
137 rt_time = time.time()
138
139 lisplay most commonly used start station
140 ular_start_station = df['Start Station'].mode()[0] # find the most popular star
141 nt('Most popular start station: %s\n' % popular_start_station)
142 lisplay most commonly used end station
143 ular_end_station = df['End Station'].mode()[0] # find the most popular end stat
144 nt('Most popular end station: %s\n' % popular_end_station)
145 lisplay most frequent combination of start station and end station trip
146 ular_trip = (df['Start Station'] + ' --> ' + df['End Station']).mode()[0] # fin
147 nt('Most popular trip: %s\n' % popular_trip)
```

棒极了

很巧妙的处理了两个站组合的问题，做的很好。建议可以再试一试直接使用groupby函数进行数据的分析。
希望下面的例子可以使你有所收获😊

```
top = df.groupby(['Start Station', 'End Station']).size().idxmax()
print("The most frequent combination of start station and end station trip is
```

```
148 nt("\nThis took %s seconds." % (time.time() - start_time))
149 nt('-'*40)
150
151 p_duration_stats(df):
152 Displays statistics on the total and average trip duration."""
153
154 nt('\nCalculating Trip Duration...\n')
155 rt_time = time.time()
156
157 lisplay total travel time
158 al = 0
159  index, row in df.iterrows():
160   total += row['duration'].total_seconds()
161 nt('\nTotal travel time: %f seconds\n' % total)
162
163 lisplay mean travel time
164 nt('Count: %d\n' % len(list(df.index.values)))
165 n_val = total / len(list(df.index.values))
166 nt('Mean travel time: %f seconds\n' % mean_val)
167
```

建议

很好的数据进行了统计，不过建议使用python的库进行数据的统计，可以简化你的程序。

```
# display total travel time
total = df['duration'].sum()
print('\nTotal travel time: %f seconds\n' % total)

# display mean travel time
mean_val = df['duration'].mean()
print('Mean travel time: %f seconds\n' % mean_val)
```

```
168
169 nt("\nThis took %f seconds." % (time.time() - start_time))
170 nt('-'*40)
171
172
173 r_stats(df):
174 Displays statistics on bikeshare users."""
175
176 nt('\nCalculating User Stats...\n')
177 rt_time = time.time()
178
179 isplay counts of user types
180 nt('\nCalculating statistics...\n\n')
181 nt('What is the breakdown of users?\n')
182 r_types = df['User Type'].value_counts()
183 nt(user_types)
184
185 isplay counts of gender
186 nt('\nCalculating statistics...\n\n')
187 nt('What is the breakdown of gender?\n')
188 'Gender' in df.columns:
189   gender = df['Gender'].value_counts()
190   print(gender)
191 e:
192   print('No gender data to share.\n')
193
194 isplay earliest, most recent, and most common year of birth
195 nt('\nCalculating statistics...\n\n')
196 nt('What is the oldest, youngest, and most popular year of birth, respectively?
197 'Birth Year' in df.columns:
```

```
198  print('Earliest year of birth: %d\n' % min(df['Birth Year'].dropna()))
199  print('Recent year of birth: %d\n' % max(df['Birth Year'].dropna()))
200  print('Most common year of birth: %d\n' % df['Birth Year'].dropna().mode()[0])
201 e:
202  print('No birth year data to share.\n')
```

棒极了

很好的考虑了有些地区数据不全的问题，做的很好。

```
203
204 nt("\nThis took %s seconds." % (time.time() - start_time))
205 nt('-'*40)
206
207
208 n():
209 le True:
210  city, month, day = get_filters()
211  df = load_data(city, month, day)
212  time_stats(df, month == 'all', day == 'all')
213  station_stats(df)
214  trip_duration_stats(df)
215  user_stats(df)
216
217  restart = input('\nWould you like to restart? Enter yes or no.\n')
218  if restart.lower() != 'yes':
219      break
220
221 me__ == "__main__":
222 n()
223
```

▶ readme.txt

返回 PATH

学员 FAQ